МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «АиСД»

Тема: Реализация идеально сбалансированного БДП

Студент гр. 8304 _____ Масалыкин Д. Р.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург

2019

Цель работы.

Изучить бинарное дерево поиска и его реализацию на языке C++, а также сбалансировать его. Реализовать операцию вставки элемента, удаления и визуализировать дерево.

Структуры данных

```
struct node
{
    int key;
    unsigned char height;
    node* left;
    node* right;
    explicit node(int k) { key = k; left = right = nullptr; height = 1; }
}
```

Структура узла дерева. Key – значение узла; height – высота дерева; left, right – указатели на левое и правое дерево; node – конструктор для структуры node.

```
unsigned char height(node* p)
```

Метод для определения высоты дерева.

```
int bfactor(node* p)
```

Фактор балансировки(разность между высотой правого и левого дерева; не должен превышать 1 по модулю)

```
void fixheight(node* p)
```

Метод для подсчета высоты.

```
node* rotateright(node* p)
node* rotateleft(node* q)
```

Методы для левого и правого поворота.

```
node* insert(node* p, int k)
```

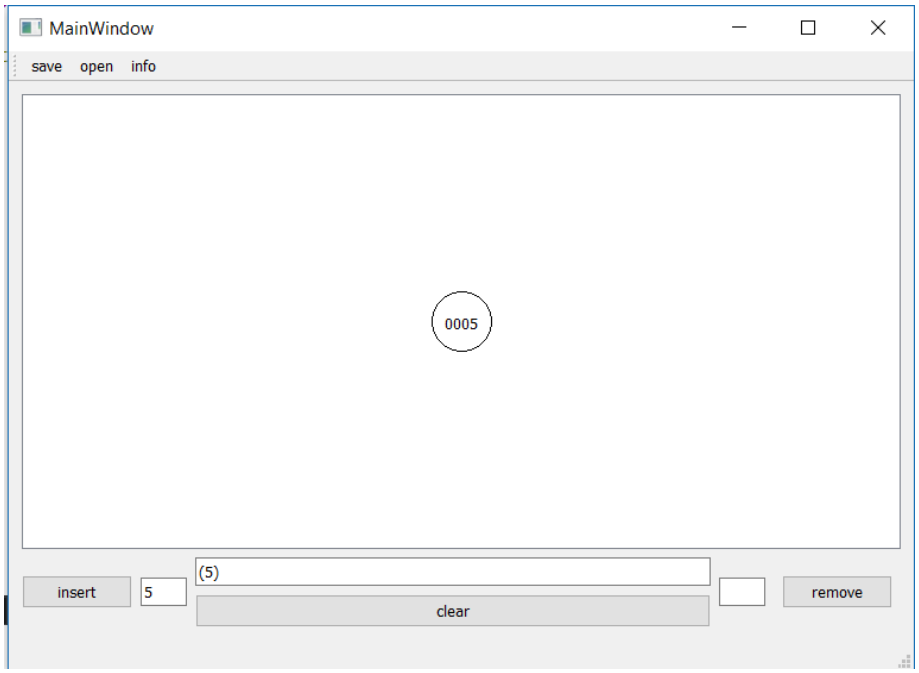
Метод для вставки ключа k в дерево p.

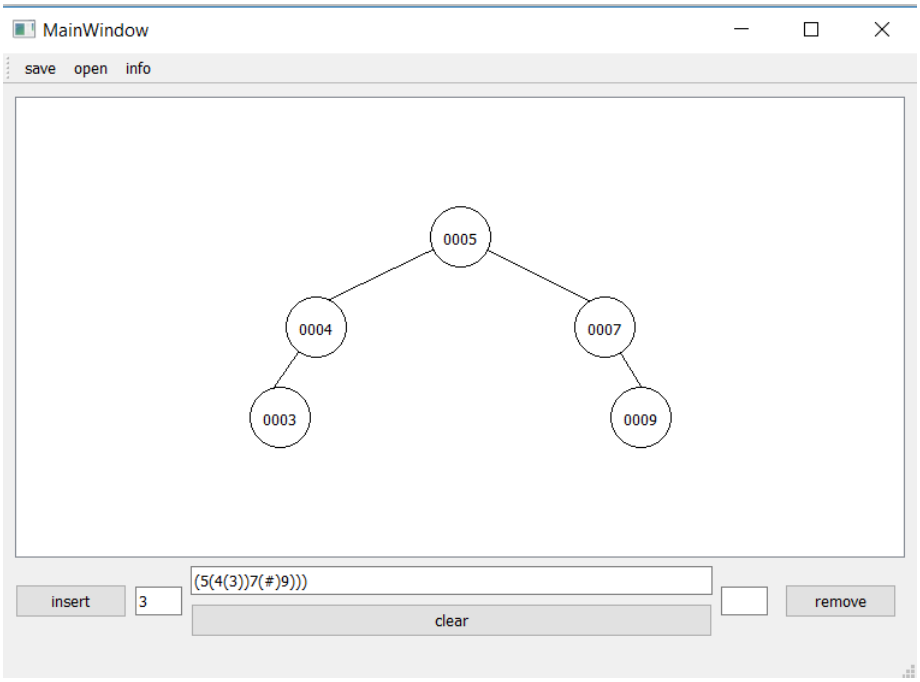
```
node* remove(node* p, int k)
```

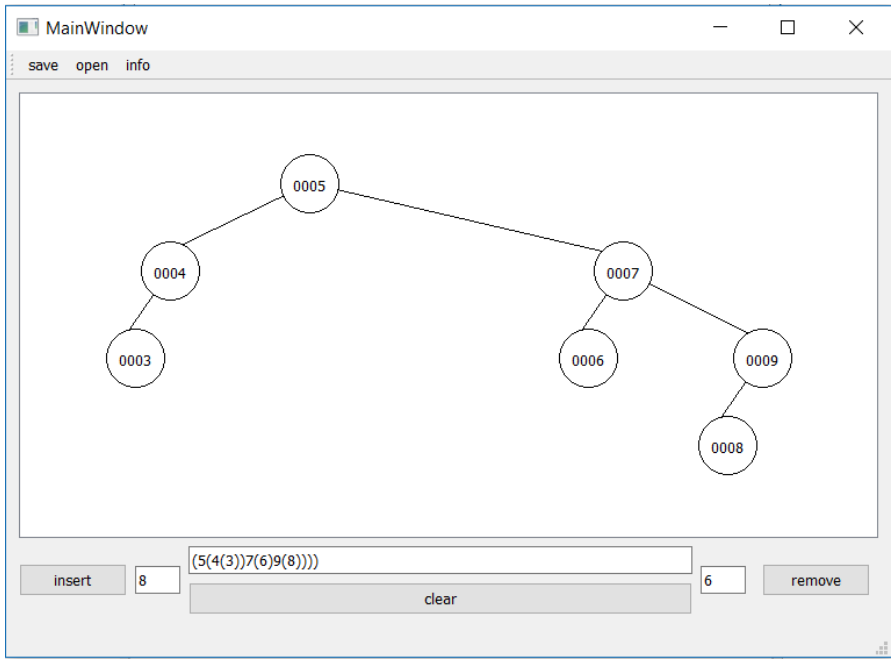
Метод для удаления ключа k из дерева p.

Тестирование программы.

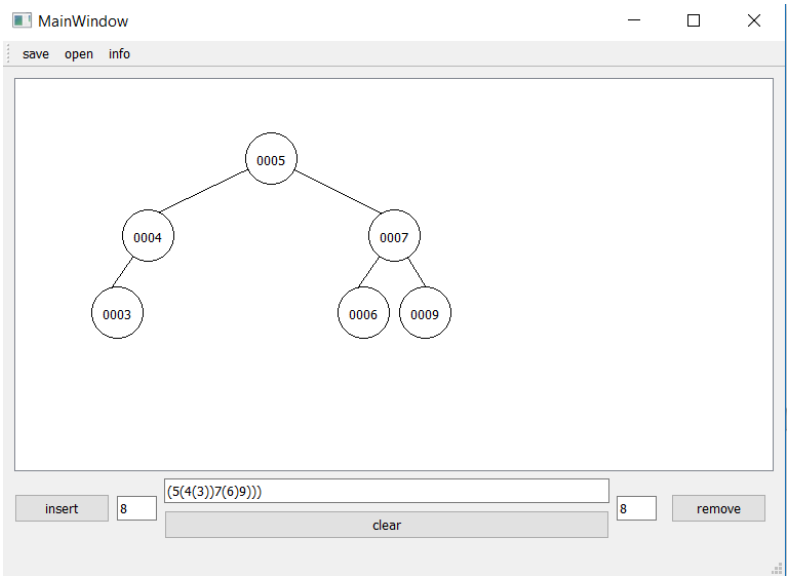
Вставка элемента.

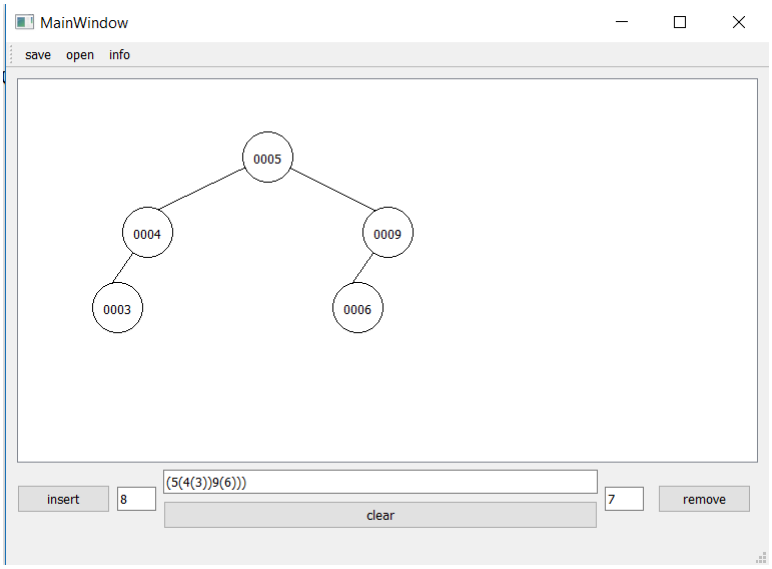

MainWindow
save
open
info
0005
(5)
insert
5
clear
remove

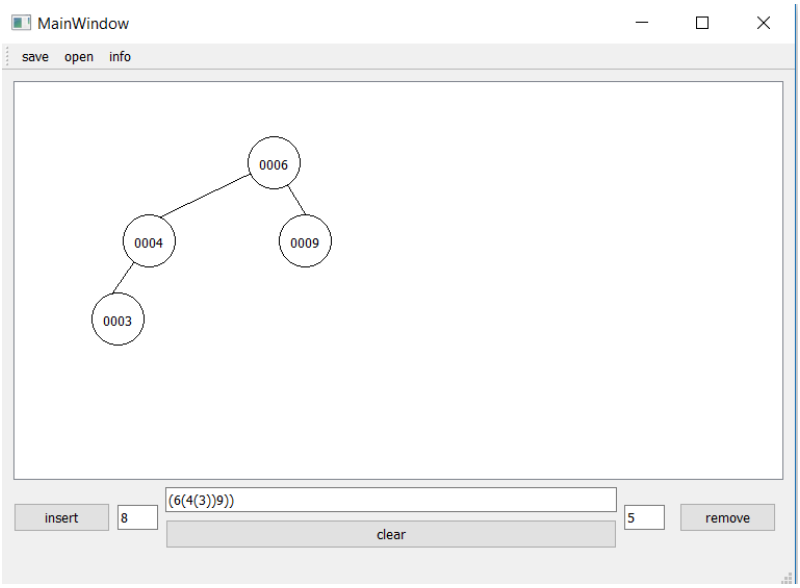

MainWindow
save
open
info
0005
0004
0007
0003
0009
(5(4(3))7(#)9)))
insert
3
clear
remove


MainWindow
save
open
info
0005
0004
0007
0003
0006
0009
0008
(5(4(3))7(6)9(8))))
insert
8
clear
6
remove

Удаление элемента.







Вывод.
В ходе работы было реализовано сбалансированное БДП со всеми основными функциями. Также была реализована визуализация при помощи средств фреймворка Qt.

Приложение А.
Исходный код программы.

Main.cpp

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->line_insert->clear();
    ui->line_remove->clear();

    scene = new QGraphicsScene(this);
    ui->visualiztion->setScene(scene);

    file.open("research.txt");
}

MainWindow::~MainWindow()
{
    delete ui;
    delete scene;
    file.close();
}

void MainWindow::on_button_insert_clicked()
{
    begin = clock();

    scene->clear();
    head = insert(head, int(stoi(ui->line_insert->text().toStdString())));

    print_tree(head, ui->visualiztion->geometry().x() / 2, 25);
    str_tree.clear();
    str_tree = "(";
    txt_tree(head, str_tree);

    ui->avl_tree->setText(QString::fromStdString(str_tree));

    end = clock();
    file<<"Insert"<<endl<<end-begin<<endl;
}

void MainWindow::on_button_remove_clicked()
{
    scene->clear();
    begin = clock();
```

```
    head = remove(head, int(stoi(ui->line_remove->text().toStdString())));
    end = clock();
    print_tree(head, ui->visualiztion->geometry().x() / 2, 25);

    file<<"Remove"<<endl<<end-begin<<endl;

    str_tree.clear();
    str_tree = "(";
    txt_tree(head, str_tree);

    ui->avl_tree->setText(QString::fromStdString(str_tree));
}

void MainWindow::on_clear_clicked()
{
    foreach (QGraphicsItem* item, scene->items()) {
          delete item;
    }

    head = delete_tree(head);
    str_tree.clear();
    str_tree = "(";

    ui->line_insert->clear();
    ui->line_remove->clear();
    ui->avl_tree->clear();

}

void MainWindow::on_actionsave_triggered()
{
    QString path = QFileDialog::getOpenFileName(this, tr("Open path to save"),
"/home/egor/Desktop");

    if (path == nullptr) return;

    ofstream file(path.toStdString());

    file << (str_tree == "(" ? "( )" : str_tree);
}

void MainWindow::on_actionopen_triggered()
{
    QString path = QFileDialog::getOpenFileName(this, tr("Open path to download tree"), "/
home/egor/Desktop");

    if (path == nullptr) return;

    ifstream oFile(path.toStdString());

    string str;
    oFile >> str;
    for (unsigned long i = 0; i < str.size(); i++)
    {
        if (str[i] == '(' || str[i] == ')' || str[i] == '#')
        {
            str[i] = ' ';
        }
    }
    stringstream sstream;
    int tmp;
```

```
    sstream << str;

    head = delete_tree(head);
    str_tree.clear();
    ui->avl_tree->clear();
    ui->line_insert->clear();
    ui->line_remove->clear();
    str_tree = "(";

    while(sstream >> tmp)
    {
        head = insert(head, tmp);
    }

    txt_tree(head, str_tree);
    ui->avl_tree->setText(QString::fromStdString(str_tree));

    print_tree(head, ui->visualiztion->geometry().x() / 2, 25);

}

void MainWindow::on_actioninfo_triggered()
{
    (new HelpBrowser (":/docs/doc", "index.htm"))->show();
}

void MainWindow::print_tree(node* tree, int x, int y)
{
    if (!tree) return;

    const int offset = 30;
    const int r = 25;

    if (tree->left)
    {
        QLine line(x + r, y + r, x - offset * tree->left->height * tree->left->height +
10, y + 90);
        scene->addLine(line, QPen(Qt::black));
    }

    if (tree->right)
    {
        QLine line(x + r, y + r, x + offset * tree->right->height * tree->right->height +
35, y + 90);
        scene->addLine(line, QPen(Qt::black));
    }

    scene->addEllipse(x, y, 2 * r, 2 * r , QPen(Qt::black), QBrush(Qt::white));

    int temp = tree->key;

    int count_zero = 1;
    while (temp /= 10)
        count_zero++;

    QString zeroes = (tree->key >= 0 ? "" : "-");
    for (int i = 0; i < 4 - count_zero; i++)
        zeroes += '0';

    QGraphicsTextItem* txtItem =
            new QGraphicsTextItem(zeroes + QString::number(abs(tree->key)));
```

```
    if (tree->key >= 0)
        txtItem->setPos(x + 7, y + 15);
    else
        txtItem->setPos(x + 4, y + 15);

    scene->addItem(txtItem);

    if (tree->left) {
        print_tree(tree->left, x - offset * tree->left->height * tree->left->height, y +
75);
   }

    if (tree->right) {
        print_tree(tree->right, x + offset * tree->right->height * tree->right->height, y
+ 75);
    }
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H

#include <QMainWindow>
#include <QFileDialog>
#include <QGraphicsScene>
#include <QGraphicsTextItem>

#include <string>
#include <fstream>
#include <ctime>
#include <fstream>

#include "avl_tree.h"
#include "help.h"

using namespace std;

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:

    void on_button_insert_clicked();

    void on_button_remove_clicked();

    void on_actionsave_triggered();

    void on_clear_clicked();
```

```
    void on_actionopen_triggered();

    void on_actioninfo_triggered();

    //void on_actionabout_triggered();

private:
    Ui::MainWindow *ui;
    //About *abt;
    QGraphicsScene* scene;

    void print_tree(node*, int, int);

    node* head = nullptr;
    QString qstr_tree = "";
    string str_tree = "(";
    time_t begin;
    time_t end;
    ofstream file;
};
#endif // MAINWINDOW_H
```

avl_tree.cpp

```
#include "avl_tree.h"

unsigned char height(node* p)
{
    return p ? p->height : 0;
}

int bfactor(node* p)
{
    return height(p->right) - height(p->left);
}

void fixheight(node* p)
{
    unsigned char hl = height(p->left);
    unsigned char hr = height(p->right);
    p->height = (hl > hr ? hl : hr) + 1;
}

node* rotateright(node* p) // правый поворот вокруг p
{
    node* q = p->left;
    p->left = q->right;
    q->right = p;
    fixheight(p);
    fixheight(q);
    return q;
}

node* rotateleft(node* q) // левый поворот вокруг q
{
    node* p = q->right;
    q->right = p->left;
    p->left = q;
    fixheight(q);
    fixheight(p);
    return p;
}
```

```
node* balance(node* p) // балансировка узла p
{
    fixheight(p);
    if( bfactor(p) == 2 )
    {
        if( bfactor(p->right) < 0 )
            p->right = rotateright(p->right);
        return rotateleft(p);
    }
    if( bfactor(p) == -2 )
    {
        if( bfactor(p->left) > 0  )
            p->left = rotateleft(p->left);
        return rotateright(p);
    }
    return p; // балансировка не нужна
}

node* insert(node* p, int k) // вставка ключа k в дерево с корнем p
{
    if( !p ) return new node(k);
    if( k < p->key )
        p->left = insert(p->left, k);
    else
        p->right = insert(p->right, k);
    return balance(p);
}

node* findmin(node* p) // поиск узла с минимальным ключом в дереве p
{
    return p->left ? findmin(p->left) : p;
}

node* removemin(node* p) // удаление узла с минимальным ключом из дерева p
{
    if( p->left == nullptr )
        return p->right;
    p->left = removemin(p->left);
    return balance(p);
}

node* remove(node* p, int k) // удаление ключа k из дерева p
{
    if( !p ) return nullptr;
    if( k < p->key )
        p->left = remove(p->left,k);
    else if( k > p->key )
        p->right = remove(p->right,k);
    else //  k == p->key
    {
        node* q = p->left;
        node* r = p->right;
        delete p;
        if( !r ) return q;
        node* min = findmin(r);
        min->right = removemin(r);
        min->left = q;
        return balance(min);
    }
    return balance(p);
}

node* delete_tree(node* tree)
{
```

```
    if (!tree) return nullptr;

    delete_tree(tree->left);
    delete_tree(tree->right);

    delete tree;

    return nullptr;
}

void txt_tree(node* tree, string& str)
{
    if (!tree)
    {
        str = "";
        return;
    }
    stringstream sstream;
    string tmp;

    sstream << tree->key;
    sstream >> tmp;

    str += tmp;

    if (tree->left)
    {
        str += "(";

        txt_tree(tree->left, str);
    } else if (tree->right)
    {
        str += "(#)";
    }
    if (tree->right)
    {
        txt_tree(tree->right, str);

    }
    str += ")";
}
```

avl_tree.h

```
#ifndef AVL_TREE_H

#define AVL_TREE_H

#include <string>
#include <sstream>
using namespace std;
struct node // структура для представления узлов дерева
{
    int key;
    unsigned char height;
    node* left;
    node* right;
    explicit node(int k) { key = k; left = right = nullptr; height = 1; }
};

unsigned char height(node* p); // определение высоты дерева
```

```
int bfactor(node* p); // баланс фактор
void fixheight(node* p); // подсчет высоты
node* rotateright(node* p); // правый поворот вокруг p
node* rotateleft(node* q); // левый поворот вокруг q
node* insert(node* p, int k); // вставка ключа k в дерево с корнем p
node* findmin(node* p); // поиск узла с минимальным ключом в дереве p
node* removemin(node* p); // удаление узла с минимальным ключом из дерева p
node* remove(node* p, int k); // удаление ключа k из дерева p
node* delete_tree(node* tree);// удаление дерева
void txt_tree(node* tree, string& str); // скобочная запись дерева

#endif // AVL_TREE_H
```

help.h

```
#ifndef HELP_H

#define HELP_H

#include <QtWidgets>

class HelpBrowser : public QWidget {
    Q_OBJECT

public:
    HelpBrowser(const QString& strPath,
                const QString& strFileName,
                      QWidget* pwgt       = nullptr
               ) : QWidget(pwgt)
    {
        QPushButton*  pcmdBack    = new QPushButton("<<");
        QPushButton*  pcmdHome    = new QPushButton("Home");
        QPushButton*  pcmdForward = new QPushButton(">>");
        QTextBrowser* ptxtBrowser = new QTextBrowser;
        setMinimumSize(400, 400);

        connect(pcmdBack, SIGNAL(clicked()),
                ptxtBrowser, SLOT(backward())
               );
        connect(pcmdHome, SIGNAL(clicked()),
                ptxtBrowser, SLOT(home())
               );
        connect(pcmdForward, SIGNAL(clicked()),
                ptxtBrowser, SLOT(forward())
               );
        connect(ptxtBrowser, SIGNAL(backwardAvailable(bool)),
                pcmdBack, SLOT(setEnabled(bool))
               );
        connect(ptxtBrowser, SIGNAL(forwardAvailable(bool)),
                pcmdForward, SLOT(setEnabled(bool))
               );

        ptxtBrowser->setSearchPaths(QStringList() << strPath);
        ptxtBrowser->setSource(QString(strFileName));

        //Layout setup
        QVBoxLayout* pvbxLayout = new QVBoxLayout;
        QHBoxLayout* phbxLayout = new QHBoxLayout;
        phbxLayout->addWidget(pcmdBack);
        phbxLayout->addWidget(pcmdHome);
        phbxLayout->addWidget(pcmdForward);
        pvbxLayout->addLayout(phbxLayout);
```

```
        pvbxLayout->addWidget(ptxtBrowser);
        setLayout(pvbxLayout);
    }
};

#endif // HELP_H
```

mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>762</width>
    <height>522</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <layout class="QGridLayout" name="gridLayout">
    <item row="1" column="1" rowspan="2">
     <widget class="QSpinBox" name="line_insert">
      <property name="buttonSymbols">
       <enum>QAbstractSpinBox::NoButtons</enum>
      </property>
      <property name="minimum">
       <number>-9999</number>
      </property>
      <property name="maximum">
       <number>9999</number>
      </property>
     </widget>
    </item>
    <item row="1" column="2">
     <widget class="QLineEdit" name="avl_tree">
      <property name="readOnly">
       <bool>true</bool>
      </property>
     </widget>
    </item>
    <item row="1" column="0" rowspan="2">
     <widget class="QPushButton" name="button_insert">
      <property name="inputMethodHints">
       <set>Qt::ImhDigitsOnly</set>
      </property>
      <property name="text">
       <string>insert</string>
      </property>
     </widget>
    </item>
    <item row="0" column="0" colspan="7">
     <widget class="QGraphicsView" name="visualiztion"/>
    </item>
    <item row="2" column="2">
     <widget class="QPushButton" name="clear">
      <property name="text">
       <string>clear</string>
```

```
    </property>
   </widget>
  </item>
  <item row="1" column="3" rowspan="2">
   <widget class="QSpinBox" name="line_remove">
    <property name="buttonSymbols">
     <enum>QAbstractSpinBox::NoButtons</enum>
    </property>
    <property name="minimum">
     <number>-9999</number>
    </property>
    <property name="maximum">
     <number>9999</number>
    </property>
   </widget>
  </item>
  <item row="1" column="5" rowspan="2">
   <widget class="QPushButton" name="button_remove">
    <property name="text">
     <string>remove</string>
    </property>
   </widget>
  </item>
 </layout>
</widget>
<widget class="QStatusBar" name="statusbar"/>
<widget class="QMenuBar" name="menubar">
 <property name="geometry">
  <rect>
   <x>0</x>
   <y>0</y>
   <width>762</width>
   <height>26</height>
  </rect>
 </property>
</widget>
<widget class="QToolBar" name="toolBar">
 <property name="windowTitle">
  <string>toolBar</string>
 </property>
 <attribute name="toolBarArea">
  <enum>TopToolBarArea</enum>
 </attribute>
 <attribute name="toolBarBreak">
  <bool>false</bool>
 </attribute>
 <addaction name="actionsave"/>
 <addaction name="actionopen"/>
 <addaction name="actioninfo"/>
</widget>
<action name="actionsave">
 <property name="text">
  <string>save</string>
 </property>
</action>
<action name="actionopen">
 <property name="text">
  <string>open</string>
 </property>
</action>
<action name="actioninfo">
 <property name="text">
  <string>info</string>
 </property>
</action>
```

```
  <action name="actionabout">
   <property name="text">
    <string>about</string>
   </property>
  </action>
 </widget>
 <resources/>
 <connections/>
</ui>
```